

## Two shaft torsion finite elements

```

% Two linear line element model of a torsional shaft loaded
% with uniform torques per length of wa on the first half
% and wb on its second half. Here the system is manually
% assembled by the (wasteful) Boolean algebra process.
%
% --> wa --> wa --> wb ----> wb ---->
% *-----(a)-----*-----(2)-----* ----> x
% 3 x=0 2J 1 x=L/2 J 2 x=L
% L=2 m, J=6.14e-7 m^4, G=7.9e10 M/m^2, wa=400 N-m/m=wb

L = 2; J = 6.14e-7; G = 7.9e10; wa = 400; wb = 400; % data

% Displacements of system: U = [U1 U2 U3]
% Mesh connection table: elem j k % node list
% a 3 1
% b 1 2
% Displacements of elements: U_a = [U3 U1], U_b = [U1 U2]
% Boolean matrix connection definitions: U_e = Boo_e * U

% Set displacement boundary condition flag and value
BC_dof = 3 ; BC_value = 0 ;

% Allocate dynamic memory
K = zeros (3, 3) % start stiffness sum at zero
F = zeros (3, 1) % start loadings sum at zero

L_e = L / 2 ; % given equal length elements
ka = 2 * G * J / L_e ; % axial stiffness of 'a'
kb = G * J / L_e ; % axial stiffness of 'b'
fa = wa * L_e ; % distributed torque on 'a'
fb = wb * L_e ; % distributed torque on 'b'

>> Boolean_shaft
K = 0 0 0
0 0 0
0 0 0
F = 0
0
0

```

## Two shaft torsion finite elements

```

Ka = 97012    -97012
    -97012    97012
Fa = 200
    200
Boo_a = 0      0      1
        1      0      0
K_a = 97012    0      -97012
        0      0      0
        -97012  0      97012
F_a = 200
        0
        200
K = 97012    0      -97012
        0      0      0
        -97012 0      97012
F = 200
        0
        200
Kb = 48506    -48506
    -48506    48506
Fb = 200
    200
Boo_b = 1      0      0
        0      1      0
K_b = 48506    -48506    0
        -48506    48506    0
        0          0      0
F_b = 200
        200
        0
K = 145518    -48506    -97012
    -48506    48506      0
    -97012     0      97012
F = 400
    200
    200

% Begin LOOP OVER ELEMENTS =====> =====> =====> ==
% Element stiffness & load arrays: K_e, F_e
Ka = ka * [ 1,  -1; ...
           -1,   1] % stiffness matrix
Fa = fa / 2 * [ 1,  1]' % load vector

% K = sum_over_e: Boo_e ^T * K_e * Boo_e
% F = sum_over_e: Boo_e ^T * F_e

% Look up its nodes in mesh table, set non-zeros
Boo_a = [0, 0, 1; ...
         1, 0, 0] % connect to nodes 3 & 1

K_a = Boo_a' * Ka * Boo_a % expand to system size
F_a = Boo_a' * Fa % expand to system size

K = K + K_a % sum rows and columns
F = F + F_a % sum rows

% Repeat LOOP OVER ELEMENTS
Kb = kb * [ 1,  -1; ...
           -1,   1] % stiffness matrix
Fb = fb / 2 * [ 1,  1]' % load vector

Boo_b = [1, 0, 0; ...
         0, 1, 0] % connect to nodes 1 & 2

K_b = Boo_b' * Kb * Boo_b % expand to system size
F_b = Boo_b' * Fb % expand to system size

K = K + K_b % sum rows and columns
F = F + F_b % sum rows

% End of matrix assembly
% End LOOP OVER ELEMENTS <===== <===== <===== <==

```

## Two shaft torsion finite elements

```

F = 400
  200
  200

K = 145518 -48506 0
    -48506 48506 0
    -97012 0 0

K = 145518 -48506 0
    -48506 48506 0
    0 0 0

K = 145518 -48506 0
    -48506 48506 0
    0 0 1

F = 400
  200
  0

U = 0.0062
    0.0103
    0

React = -800.0000

U_a = 0
      0.0062

Ra = -800.0000
     400.0000

e_a = 0.0062

s_a = 4.8860e+08

U_b = 0.0062
      0.0103
Rb = -400.0000
     -0.0000

e_b = 0.0041

s_b = 3.2573e+08

% K * U = F minimizes energy, but does not yet satisfy
%       the essential displacement boundary condition

% Save BC row(s) for later system reaction recovery
BC_row = K (BC_dof, 1:3); BC_F = F(BC_dof);% or partition eqs

% Enforce displacement boundary condition(s)
F = F - BC_value * K (1:3, BC_dof) % move knowns to RHS

% SOLVE MATRIX EQUILIBRIUM EQUATIONS, with unique BC
% partition the matrices or trick them, here use trick
K (1:3, BC_dof) = 0 % zero known column
K (BC_dof, 1:3) = 0 % UBC_dof = BC_value identity
K (BC_dof, BC_dof) = 1 % diagonal 1 in blank row, col
F (BC_dof) = BC_value % identity done, end trick

% Solve system equations, which now are non-singular
U = K \ F % All displacements known. Post-process elements

% Recover the system reaction(s). (Or use partitioned eqs.)
React = BC_row * U - BC_F

% Begin LOOP OVER ELEMENTS =====> =====> =====> =====> =====>
% Extract element displacements
U_a = Boo_a * U % get results 3 & 1

% Get element end force reactions (optional)
Ra = Ka * U_a - Fa % two forces external to element

% Get element (centroid) strains
e_a = [-1 1] * U_a / L_e % mid-length strain

% Get element (centroid) stresses
s_a = G * e_a % mid-length stress

% Repeat LOOP OVER ELEMENTS
U_b = Boo_b * U % get results 1 & 2
Rb = Kb * U_b - Fb % two forces external to element
e_b = [-1 1] * U_b / L_e % mid-length strain
s_b = G * e_b % mid-length stress

% End LOOP OVER ELEMENTS <===== <===== <===== <===== <=====

```

## Two shaft torsion finite elements

```
>> addpath /net/course-a/mech517/public_html/Matlab_Plots

>> mesh_shrink_plot
Read 3 mesh coordinate pairs
Read 2 elements with 2 nodes each

>> result_1d_graph(1)
Read 3 mesh coordinate pairs
Read 2 elements connections
Read 1 nodal solution values
    with 3 components each
Max value is 0.0062 at node 1
Min value is 0.0062 at node 1
>> quit
% more *.tmp
::::::::::::::::::
msh_bc_xyz.tmp
::::::::::::::::::
0  1.0
0  2.0
1  0.0
::::::::::::::::::
msh_typ_nodes.tmp
::::::::::::::::::
1  3  1
1  1  2
::::::::::::::::::
node_results.tmp
::::::::::::::::::
0.0062
0.0103
0
```

